1)Push and Pop Operations

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
struct Stack{
    int size;
    int top;
    int *arr;
};

void push(struct Stack *s,int element){
    if(s->top==s->size-1){
        printf("Stack Overflow\n");
    }
    else{
        s->top++;
        s->arr[s->top]=element;
        printf("%d inserted\n",element);
    }
}
int pop(struct Stack *s){
    int t;
    if(s->top==-1){
        printf("Stack Underflow\n");
    }
    else{
        t=s->arr[s->top];
        s->top--;
        printf("%d popped out\n",t);
        return t;
    }
}
int main(){
    struct Stack s;
    printf("Enter the length of array : ");
    scanf("%d",&s.size);
    s.arr=(int*)malloc(s.size*sizeof(int));
    s.top=-1;
    int choice,ele;
    bool again= true;
```

```c
    bool again= true;
    while(again){
        printf("Enter 1 to push\n");
        printf("Enter 2 to pop\n");
        printf("Enter 0 to stop\n");
        scanf("%d",&choice);
        switch (choice)
        {
        case 1:
            printf("Enter a element : ");
            scanf("%d",&ele);
            push(&s,ele);
            break;
        case 2:
            pop(&s);
            break;
        case 0:
            again=false;
            break;
        default:
            break;
        }
    }
    return 0;
}
```

```
PS D:\ENGINEERING\DSA_C\PRAC_2> cd "d:\ENGINEERING\DSA
Enter the length of array : 2
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
1
Enter a element : 5
5 inserted
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
1
Enter a element : 8
8 inserted
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
1
Enter a element : 6
Stack Overflow
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
2
8 popped out
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
2
5 popped out
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
2
Stack Underflow
Enter 1 to push
Enter 2 to pop
Enter 0 to stop
0
PS D:\ENGINEERING\DSA_C\stack>
```

2)Infix to Postfix and Solving postfix Expression

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Stack {
    int size;
    int top;
    char *arr;
};
int stackTop(struct Stack *sp){
    return sp->arr[sp->top];
}

int isEmpty(struct Stack *s){
    if(s->top==-1){
        return 1;
    }
    return 0;
}
int isFull(struct Stack *s){
    if(s->top==s->size-1){
        return 1;
    }
    return 0;
}

void push(struct Stack *s,int element){
    if(isFull(s)){
        printf("Stack Overflow");
    }
    else{
        s->top++;
        s->arr[s->top]=element;
    }
}

char  pop(struct Stack *s){
    char t;
    if(isEmpty(s)){
        printf("Stack Underflow");
    }
    else{
        t=s->arr[s->top];
        s->top--;
        return t;
    }
}

int precedence(char ch){
    if(ch=='*'|| ch=='/'){
        return 3;
    }
    else if(ch=='+'|| ch=='-'){
        return 2;
    }
    return 0;
}

int isOperator(char ch){
    if(ch=='+' ||ch=='-' ||ch=='*' ||ch=='/' ){
        return 1;

    }
    return 0;
}
```

```c
char *infixToPostfix(char *infix){
    struct Stack *sp=(struct Stack *)malloc(sizeof(struct Stack)) ;
    sp->size=100;
    sp->top=-1;
    sp->arr=(char*)malloc(sp->size*sizeof(char));
    char *postfix=(char*)malloc((strlen(infix)+1)*sizeof(char));
    int i=0,j=0;
    while(infix[i]!='\0'){
        if(!isOperator(infix[i])){
            postfix[j]=infix[i];
            i++;j++;
        }
        else{
            if(precedence(infix[i])>precedence(stackTop(sp))){
                push(sp,infix[i]);
                i++;
            }
            else{
                postfix[j]=pop(sp);
                j++;
            }
        }
    }
    while(!isEmpty(sp)){
        postfix[j]=pop(sp);
        j++;
    }
    postfix[j]='\0';
    return postfix;
}
int solve(char *eqn){
    int len=strlen(eqn);
    struct Stack *sp;
    sp->top=-1;
    sp->arr=(char*)malloc(sizeof(char)*11);
    for (int i = 0; eqn[i]!='\0'; i++)
    {
        if(isdigit(eqn[i])){
            push(sp,eqn[i]-48);
        }
        else if(isOperator(eqn[i])==1){
            int var2=pop(sp);
            int var1=pop(sp);
            switch (eqn[i])
            {
            case '+':
                push(sp,var1+var2);
                break;
            case '-':
                push(sp,var1-var2);

                break;
            case '/':
                push(sp,var1/var2);

                break;
            case '*':
                push(sp,var1*var2);

                break;

            default:
                break;
            }
        }
    }  return pop(sp);
}
int main()
{

    char *infix="8+8/4+3";
    char *postfix=infixToPostfix(infix);
    printf("Postfix eqn : %s\n",postfix);
    printf("Ans : %d",solve(postfix));
    return 0;}
```

```
PS D:\ENGINEERING\DSA_C\
Infix eqn : 8+8/4+3
Postfix eqn : 884/+3+
Ans : 13
```